Отчёт по лабораторной работе №10

*дисциплина: Архитектура компьютера*

Максим Александрович Мишонков

Содержание

# 1 Цель работы

Целью данной лабораторной работы является приобретение навыков написания программ с использованием подпрограмм, знакомство с методами откладки при помощи GDB и его основными возможностями.

# 2 Задание

Научиться писать программы с использованием подпрограмм, ознакомиться с методами откладки при помощи GDB и его основными возможностями.

# 3 Теоретическое введение

**Откладка** - это процесс поиска и исправления ошибок в программе (обнаружение ошибки, поиск её местонахождения, определение причины ошибки, исправление ошибки)

**Точки останова** - это специально отмеченные места в программе, в которых программа-откладчик приостанавливает выполнение программы и ждёт команд.

Откладчик GDB позволяет увидеть, что происходит “внутри” программы в момент её выполнения и что делает программа в момент сбоя. GDB может начать выполнение программы, задав всё, что может повлиять на её поведение; остановить программу при указанных условиях; исследовать, что случилось, когда программа остановилась; изменить программу так, чтобы можно было поэкспериментировать с устранением эффектов одной ошибки и продолжить выявление других.

**Подпрограмма** - это, как правило, функционально законченный участок кода, который можно многократно вызывать из разных мест программы.

Если в подпрограмме встречается одинаковый участок кода, его можно оформить в виде подпрограммы, а во всех нужных местах поставить её вызов. При этом подпрограмма будет содержаться в коде в одном экземпляре, что позволит уменьшить размер кода всей программы.

Для вызова подпрограммы из основной программы применяется инструкция call, которая заносит адрес следующей инструкции в стек и загружает в регистр eip адрес соответствующей подпрограммы. Завершается подпрограмма инструкцикй ret, которая извлекает из стека адрес, занесённый туда соответствующей инструкцией call, и заносит его в eip.

# 4 Выполнение лабораторной работы

1. Создал каталог для выполнения лабораторной работы №10, перешёл в него и создал файл lab10-1.asm. (рис. 1)

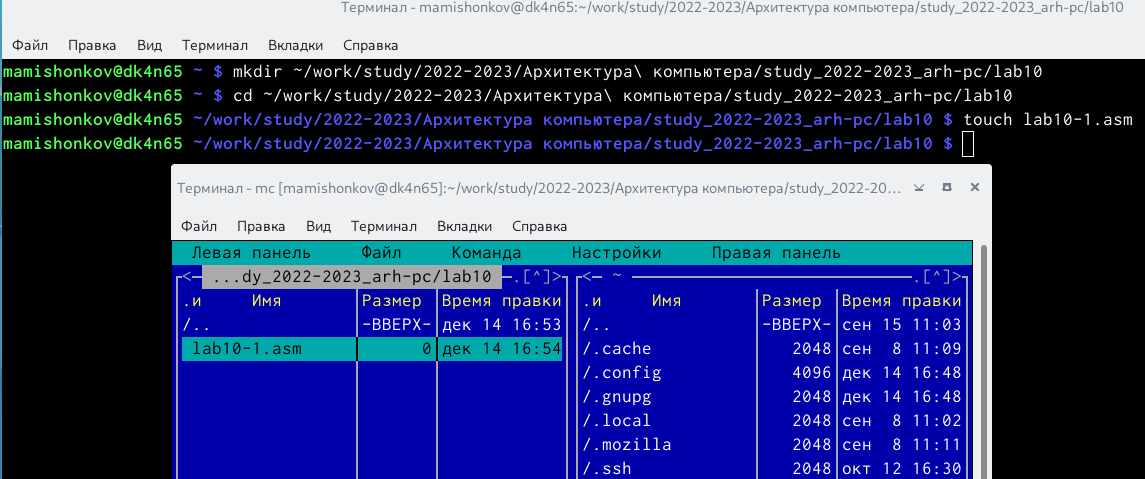


Рис. 1: Создание каталога и файла

1. Ввёл в файл lab10-1.asm текст программы из листинга 10.1. (рис. 2)

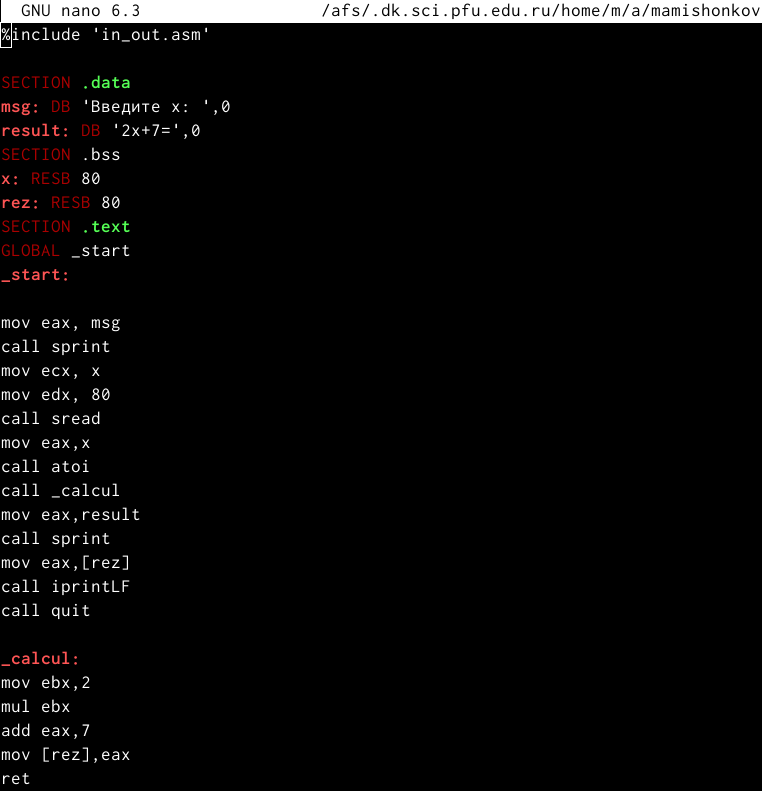


Рис. 2: Введение текста программы

1. Создал исполняемый файл и проверил его работу. (рис. 3)

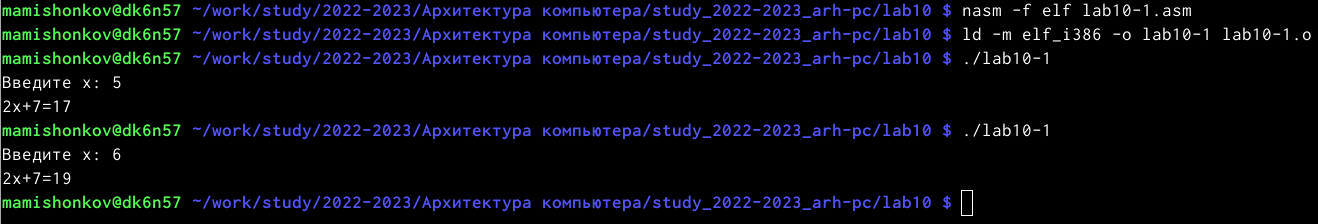


Рис. 3: Проверка работы исполняемого файла

1. Изменил текст программы, добавив подпрограмму \_subcalcul в подпрограмму \_calcul, для вычисления выражения f(g(x)), где x вводится с клавиатуры, f(x)=2x+7, g(x)=3x-1. Проверил работу исполняемого файла. (рис. 4, 5, 6)

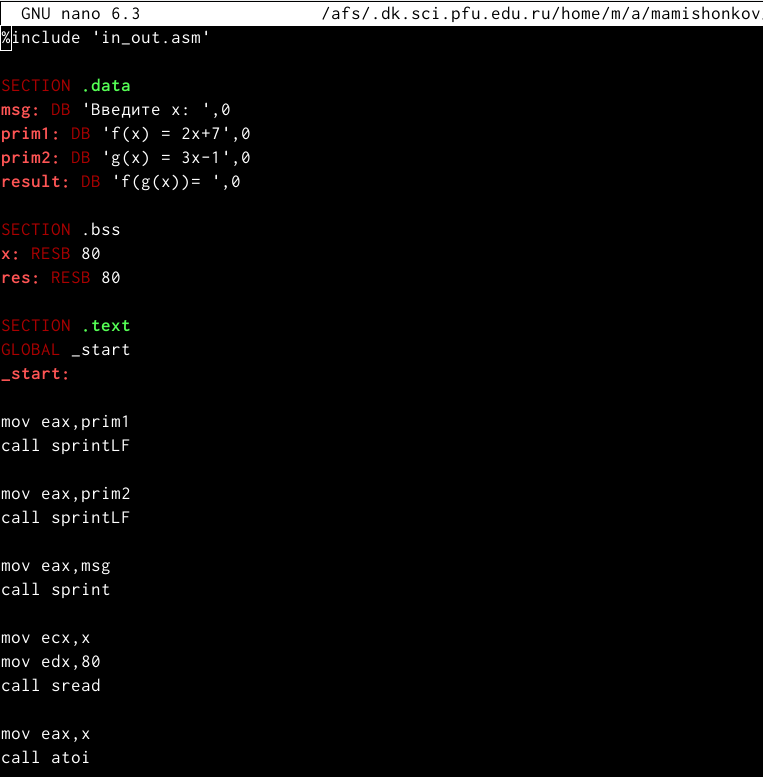


Рис. 4: Изменение текста программы

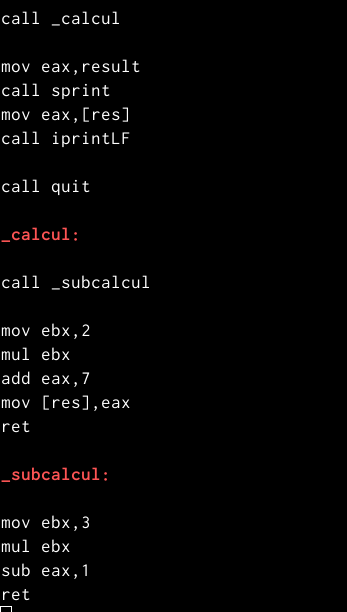


Рис. 5: Изменение текста программы

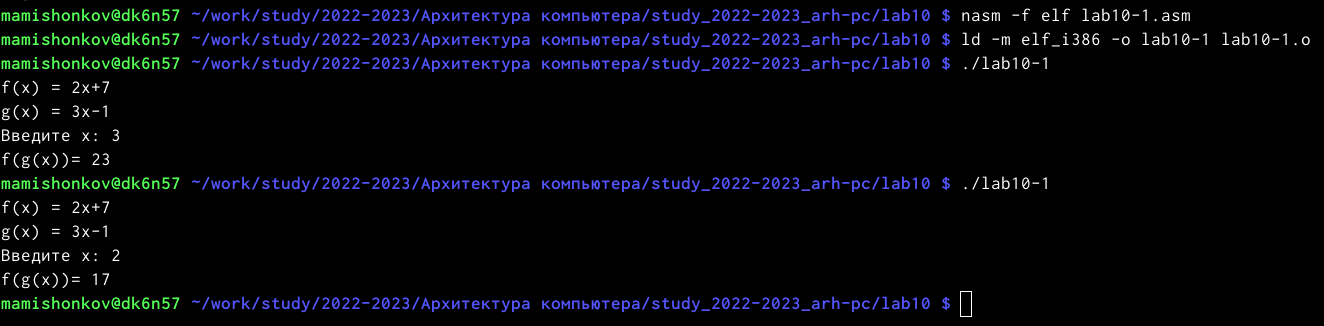


Рис. 6: Проверка работы исполняемого файла

1. Создал файл lab10-2.asm. (рис. 7)

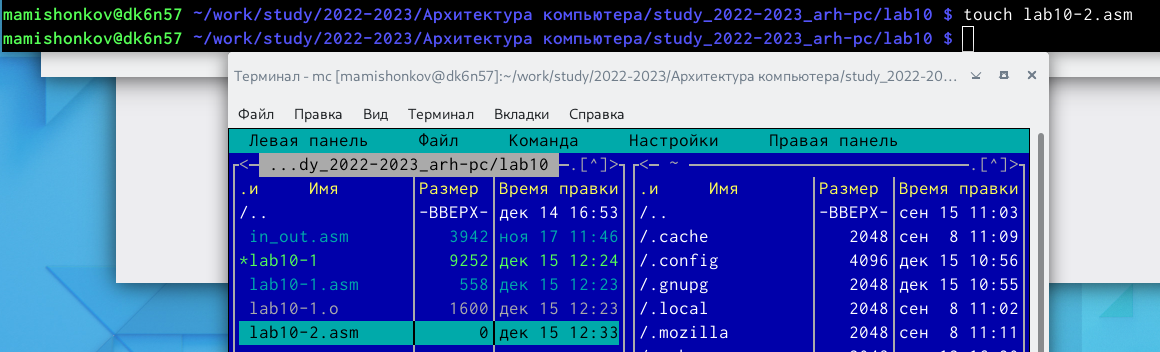


Рис. 7: Создание файла

1. Ввёл в файл lab10-2.asm текст программы из листинга 10.2. (рис. 8)



Рис. 8: Введение текста программы

1. Получил исполняемый файл. Для работы с GDB в исполняемый файл добавил отладочную информацию, проведя трансляцию программ с ключом ‘-g’. Загрузил исполняемый файл в отладчик gdb. Проверил работу программы, запустив её в оболочке GDB с помощью команды run. (рис. 9)

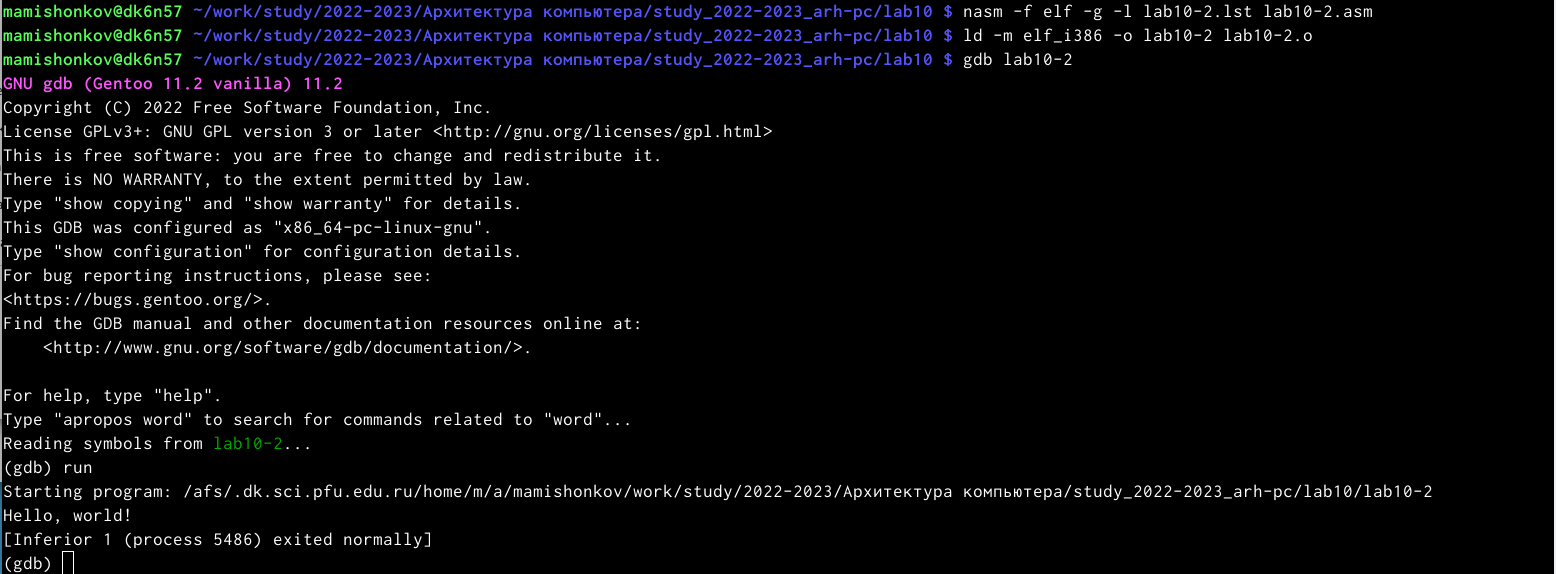


Рис. 9: Проверка работы исполняемого файла

1. Для более подробного анализа программы установил брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запустил её. (рис. 10)

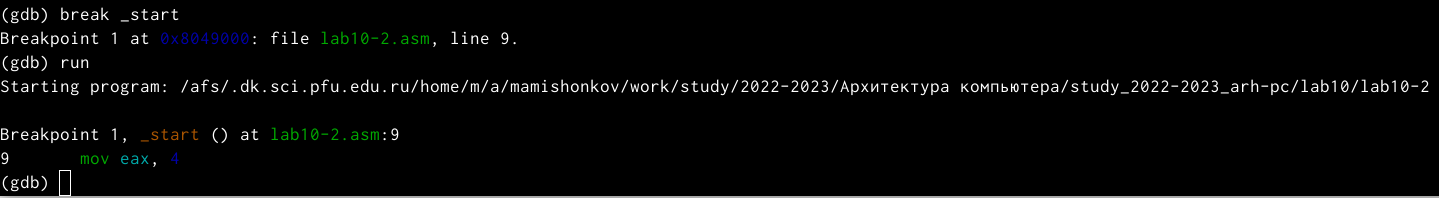


Рис. 10: Запуск программы

1. Посмотрел дисассимилированный код программы с помощью команды disassemble, начиная с метки \_start. (рис. 11)

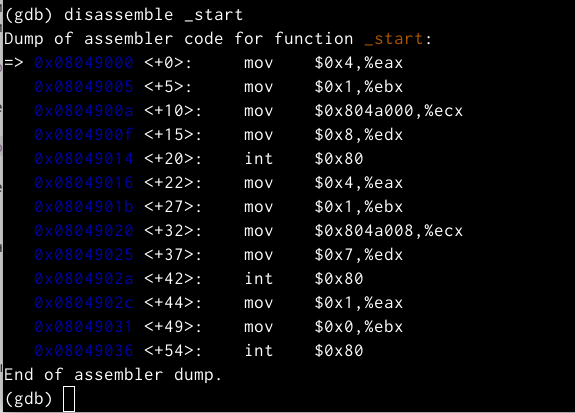


Рис. 11: Просмотр дисассимилированного кода программы

1. Переключился на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel. (рис. 12)

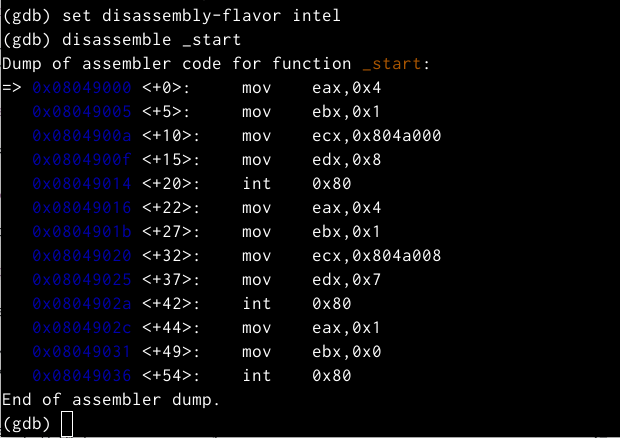


Рис. 12: Переключение на отображение команд с Intel’овским синтаксисом

1. Включил режим псевдографики для более удобного анализа программы. (рис. 13, 14 )

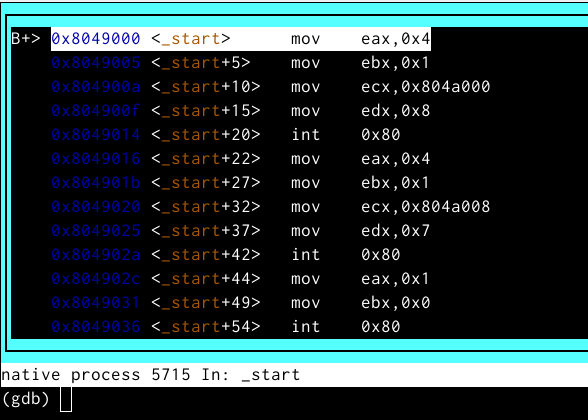


Рис. 13: Включение режима псевдографики

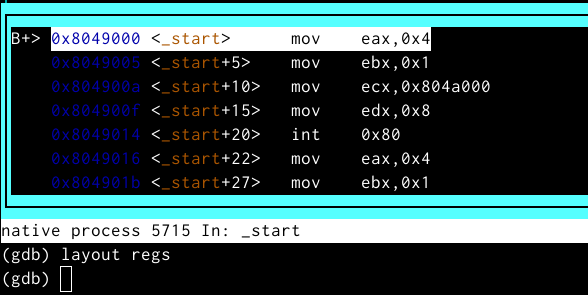


Рис. 14: Включение режима псевдографики

1. На предыдущих шагах была установлена точка останова по имени метки (\_start). Проверил это с помощью команды info breakpoints. (рис. 15)

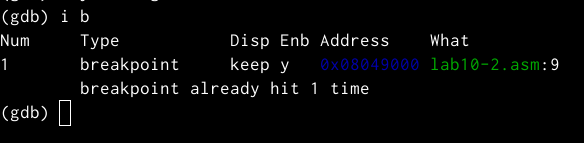


Рис. 15: Проверка точки останова по имени метки (\_start))

1. Установил ещё одну точку останова по адресу инструкции, определив адрес предпоследней инструкции. (рис. 16)

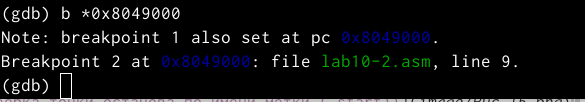


Рис. 16: Установка ещё одной точки останова по адресу инструкции

1. Посмотрел информацию о всех установленных точках установа. (рис. 17)

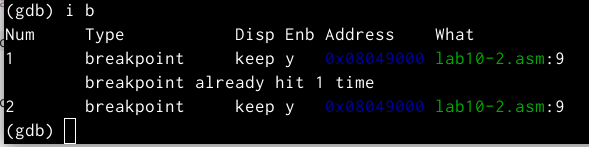


Рис. 17: Просмотр информации о всех установленных точках установа

1. Посмотрел содержимое регистров с помощью команды info registers. (рис. 18, 19)

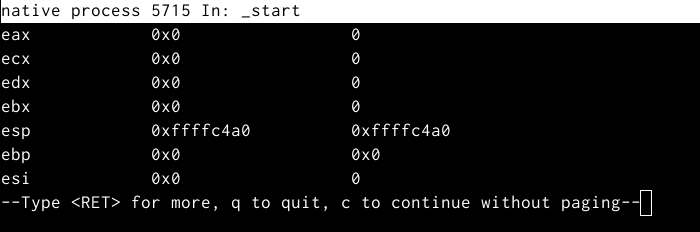


Рис. 18: Просмотр содержимого регистров

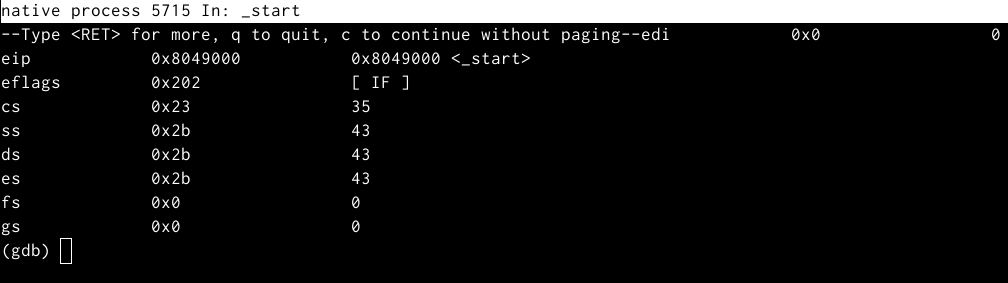


Рис. 19: Просмотр содержимого регистров

1. Посмотрел значение переменной msg1. (рис. 20)

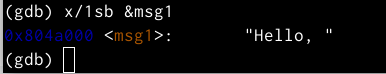


Рис. 20: Значение переменной msg1

1. Посмотрел значение переменной msg2. (рис. 21)

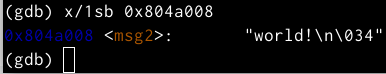


Рис. 21: Значение переменной msg2

1. Изменил первый символ переменной msg1. (рис. 22)

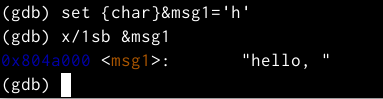


Рис. 22: Изменение первого символа переменной msg1

1. Заменил один из символов во второй переменной msg. (рис. 23)

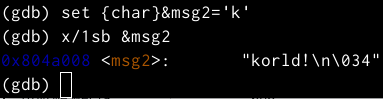


Рис. 23: Замена символа во второй переменной msg

1. Вывел в различных форматах значение регистра edx. (рис. 24)

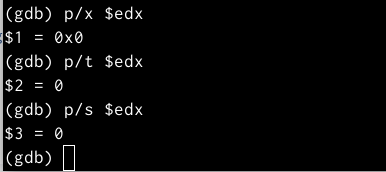


Рис. 24: Вывод в различных форматах значения регистра edx

1. C помощью команды set изменил значение регистра ebx. (рис. 25)

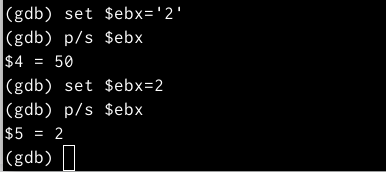


Рис. 25: Изменение значения регистра ebx

1. Скопировал файл lab9-2.asm, созданный при выполнении лабораторной работы №9, с программой, выводящей на экран аргументы командной строки, в файл с именем lab10-3.asm. (рис. 26)

Рис. 26: Копирование файла lab9-2.asm в файл lab10-3.asm

Рис. 26: Копирование файла lab9-2.asm в файл lab10-3.asm

1. Создал исполняемый файл, загрузил его в отладчик, указав аргументы. (рис. 27)

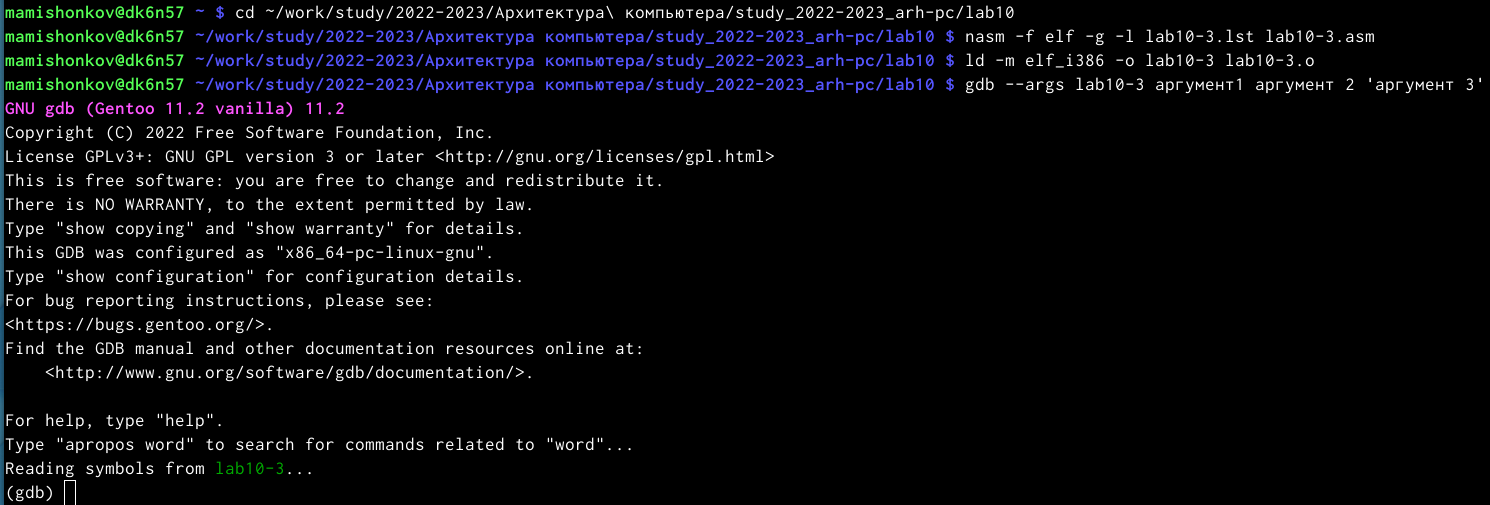


Рис. 27: Создание исполняемого файла и загрузка его в отладчик

1. Установил точку останова перед первой инструкцией в программе и запустил программу. (рис. 28)

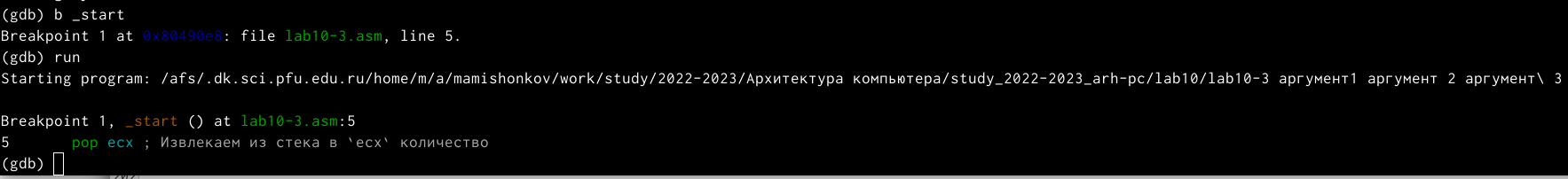


Рис. 28: Установка точки останова перенд первой инструкцией в программе и её запуск

1. Адрес вершины стека хранится в регистре esp и по этому адресу располагается число, равное количеству аргументов командной строки (включая имя программы). Как видно, число аргументов равно 5. Это имя программы lab10-3.asm. (рис. 29)

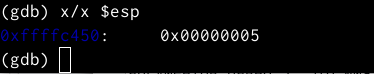


Рис. 29: Команда x/x $esp

1. Просмотрел остальные позиции стека. По адресу esp+4 располагается адрес в памяти, где находится имя программы, по адресу esp+8 хранится адрес первого аргумента, по адресу esp+12 - второго аргумента и т.д. (рис. 30)

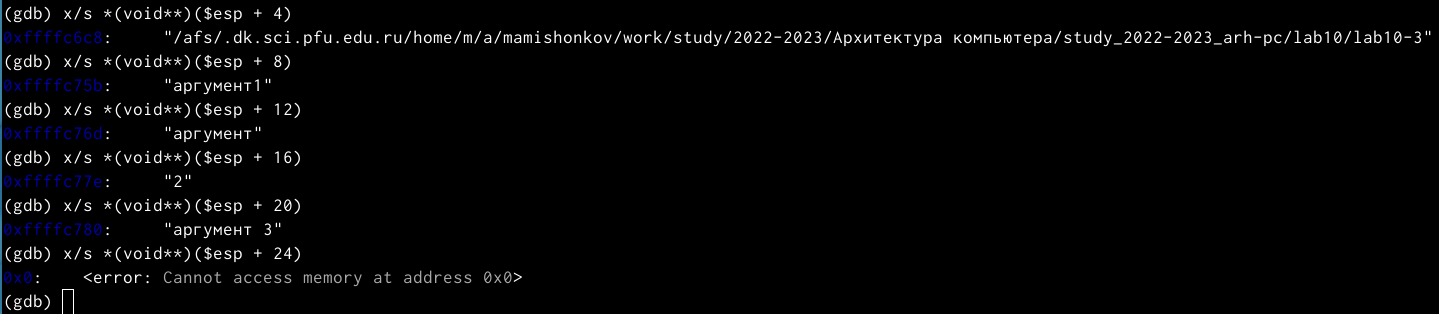


Рис. 30: Просмотр остальных позиций стека

**Самостоятельная работа**

1. Преобразовал программу из лабораторной работы №9 (задание 1 из самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму. Создал исполняемый файл и проверил его работу. (рис. 31, 32, 33)



Рис. 31: Преобразование программы

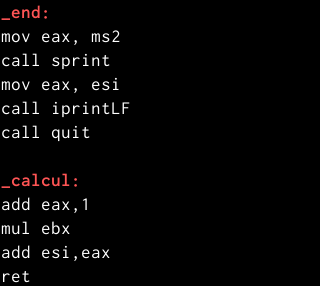


Рис. 32: Преобразование программы

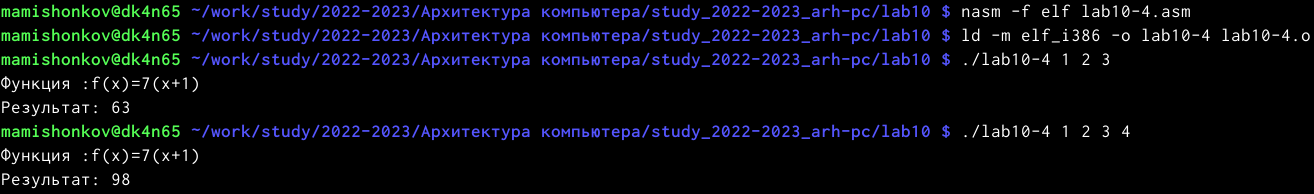


Рис. 33: Проверка работы исполняемого файла

1. C помощью отладчика GDB, анализируя изменения значений регистров, определил и исправил ошибку в программе вычисления выражения (3+2)\*4+5. В результате программа должна выглядеть следующим образом. Создал исполняемый файл и проверил его работу. (рис. 34, 35)

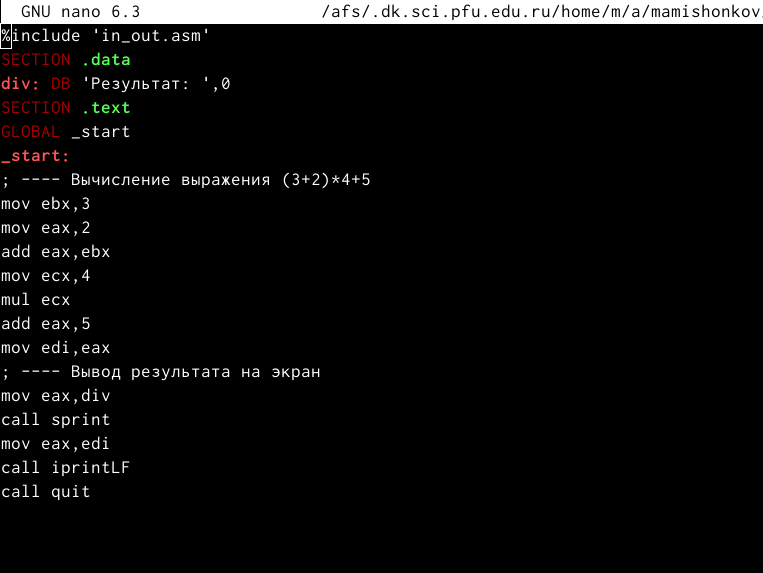


Рис. 34: Изменение текста программы

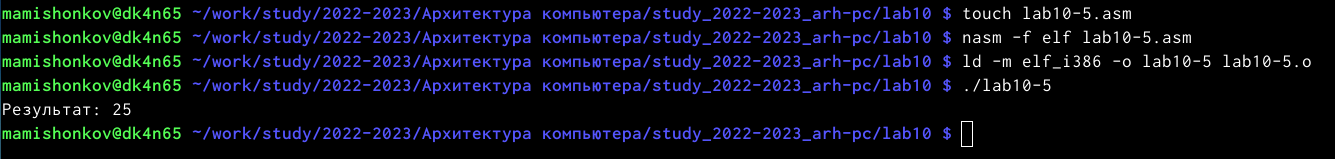


Рис. 35: Проверка работы исполняемого файла

# 5 Выводы

В ходе выполнения данной лабораторной работы я научился писать программы с использованием подпрограмм, ознакомился с методами откладки при помощи GDB и его основными возможностями.